Republic of the Philippines
CAMARINES SUR POLYTECHNIC COLLEGES
Nabua, Camarines Sur

COLLEGE *of* COMPUTER STUDIES

CCCS 104 - Data Structures and Algorithms

LEARNING TASK (LINEAR DATA STRUCTURE - STACK)

NAME: ***JAYSON ASIADO*** SECTION: ***BSCS 2A***

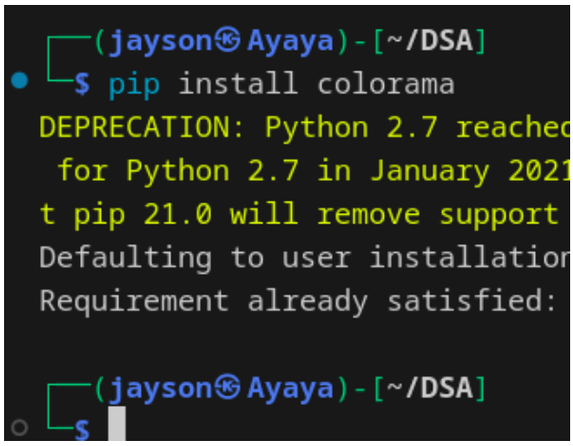
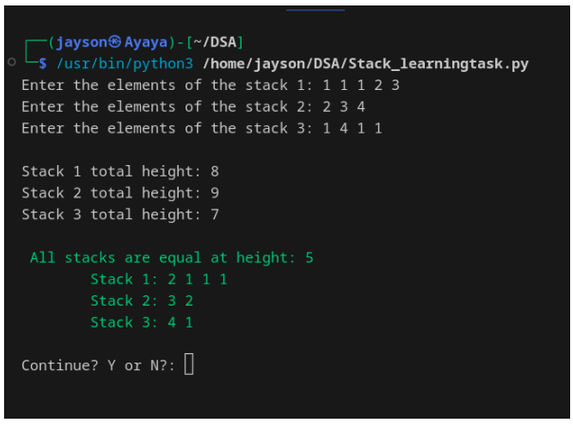
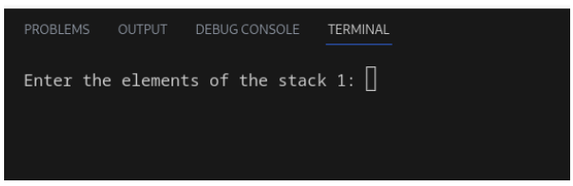
RATIONALE

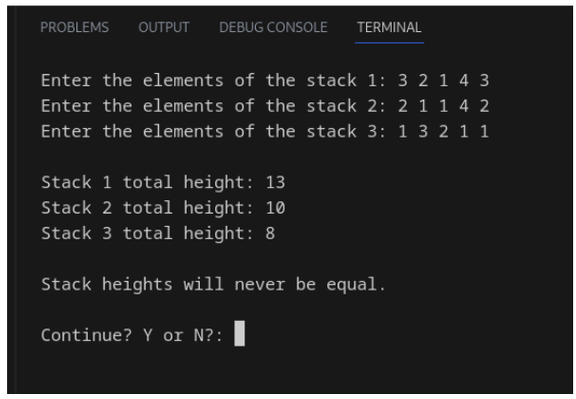
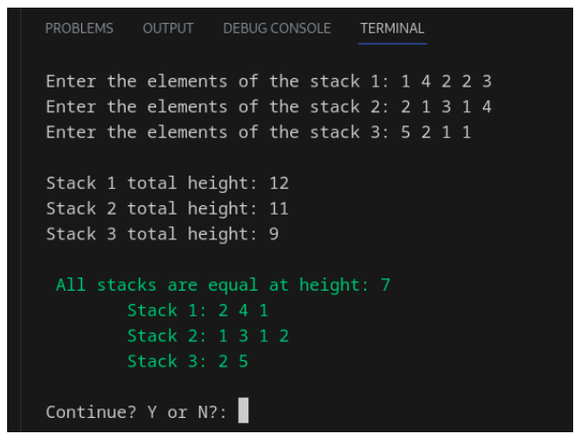
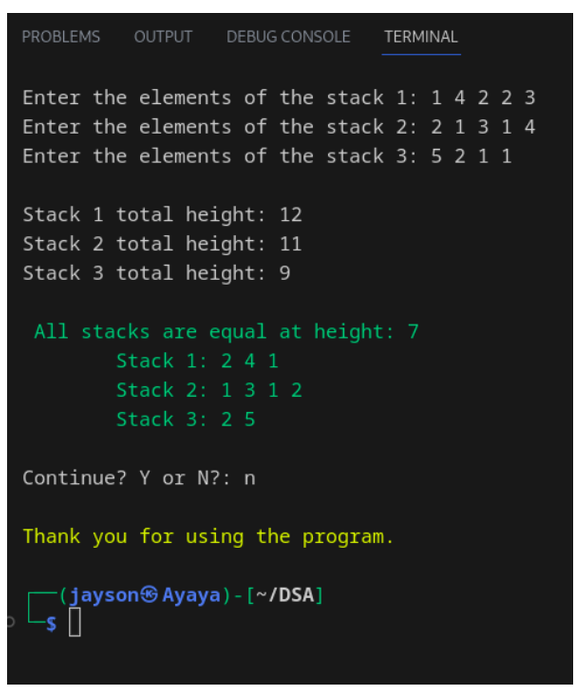
Linear Stack Data Structure is a data structure that uses the term FILO(first in last out) where the first data you insert is the last will go out since you can't access the data in a deep position without first removing the data above it. Stack works only in two operations: the Push and Pop. Pop is only exclusive at the top of the stack since you can't specify the position where you can add the data. Pop is an operation where it will delete the last data you insert at the stack. Example of a Stack is the task scheduling of an operating system. Task scheduling is the same as stack data structure because an operating system processes the top or the currently running program and does not go to another process unless you exit or the program is fully executed.

My written Program is a Stack linear data structure that operates Push and Pop operation but this program ask the user for three stack with elements separated by whitespaces and store it to different variable the stack1,stack2, and stack3 after that, the program finds the height where all the sum of the stacks are equal by popping the last element of the stack with the highest sum until they're all equal. However, if the program didn't find an equal height after all those popping, a message prompt will show "The Stack will never be equal" saying that there's no equal height in the stacks. But if the program finds an equal height it will return the number that the sum of the stack is equal.

USER GUIDE

Step by step instructions on how to use your program. Include images for easily visualization

Step 1 *Before running the program, first install the library called colorama. I used colorama to add some color to make it just fancy and presentable. To install, just go to the terminal and type; "pip install colorama" if you already installed this library, you don't need to install it again. In my case, It says the requirement is already satisfied. It means I already installed it and it's good to go.*	*Image 1* 
Step 2 *Tap the run button until you see the prompt that will ask you for the first stack. In this example I will be using the examples given at the LeOnS. So now it will ask you for the 3 stack that you want to add and I just added the 3 stack in the example. Once I press enter, it will show the total height of each stack and will show next if the stack will equal at the specific height. As you can see, The program says that the stacks are equal at height 5 and will print the remaining elements.*	*Image 2* 
Step 3 *Since we have 2 more examples we are going to type "y" to continue the program. Don't worry we're going to choose the 'n' at the end of these steps so for now I type y and it will clear all the text in the*	*Image 3* 

terminal and ready for the next 3 stack inputs from us.	
Step 4 *And after putting all the elements on the 3 stacks, we need to press enter again to see the result. But now it says The stack will never be equal and that's true because the elements will be empty if the heights are not equal to each other so we received a prompt message that says the stacks will never be equal.*	*Image 4* 
Step 5 *I type again "y" to continue the program since we still have our last example.* *I enter all the 3 stacks and just like before, once I press enter, It will show the total height of each stack and it shows that the stack are all equal at height 7 and prints the remaining elements of the stack.*	*Image 5* 
Step 6 *This is the last step since we don't have more examples left but feel free to put some inputs to see the different results.* *And now that we are done with the program I'm going to type 'n' to end or exit the program and as soon as I press enter, It says "Thank you for using the program". That's it we are done running the program and adding some stacks and saw some different results. But if you want to run the program again just hit the run button again and it will run again just like the first step.*	*Image 6* 

PROGRAM CODE

```
"""
JAYSON ASIADO BSCS 2A
"""
import os
from colorama import Fore,Style
def create_stack(): # I create an empty stack that I can use to push and
pop elements
   return []

def push(stack,value): # this is the push function that I can use to add
elements to the stack
   return stack.append(value)  # append the value to the last index of
the stack

def is_empty(stack): # Function for cheking if the stack is empty
   return stack == 0 # return true if the stack is equal to 0

def pop(stack):# pop function is to remove the last element of the stack
   return stack.pop() if stack is not is_empty(stack) else "The stack is
empty" # just pop the last element of the stack if the stack is not
empty

def user_input(): # this function is for getting the user input
   stack = create_stack() # create a stack
   for i in range(1,4): # loop 3 times to get the elements of the stack
       # I used the map function to convert the input to integer and
split the input by space
       # push every element to the stack by using the push function and
strip white spaces in the input
       push(stack,(list(map(int,input(f"Enter the elements of the stack
{i}: ").strip().split()))))
   print(f"\nStack 1 total height: {sum(stack[0])}\nStack 2 total
height: {sum(stack[1])}\nStack 3 total height: {sum(stack[2])}")
   return stack # return the stack

def equal_height(stack1,stack2,stack3): # this function is for checking
if the stacks are going to be equal after popping elements
   # I used the sum function to get the sum of the elements of the stack
   stack1sum = sum(stack1)
   stack2sum = sum(stack2)
   stack3sum = sum(stack3)
```

```
    while 1: # I used the while loop and pop an item for every iteration
in the maximum sum of the stacks
        if stack1sum == 0 or stack2sum == 0 or stack3sum == 0: # if the
sum of the stacks is equal to 0 then return the stack willl never be
equal
            return print("\nStack heights will never be equal.")
        if stack1sum == stack2sum == stack3sum: # This is to check if the
stacks are equal then return any of the sum
            return print(Fore.GREEN+f"""\n All stacks are equal at
height: {stack1sum}
        Stack 1: {' '.join(map(str,stack1[::-1]))}
        Stack 2: {' '.join(map(str,stack2[::-1]))}
        Stack 3: {' '.join(map(str,stack3[::-1]))}"""+Style.RESET_ALL)#
print the reverse of the remaining stack

        max_sum = max(stack1sum,stack2sum,stack3sum) # get the maximum
sum of the stacks

        if stack1sum == max_sum: # if the stack1sum is equal to the
maximum sum and the index is less than the length of the stack
            stack1sum -= stack1[-1] # subtract the last element of the
stack to the stack1sum
            pop(stack1) # pop the last element of the stack
            # JUST LIKE THE STACK1 I DID THE SAME THING TO THE STACK2 AND
STACK3
        elif stack2sum == max_sum:
            stack2sum -= stack2[-1]
            pop(stack2)
        else:
            stack3sum -= stack3[-1]
            pop(stack3)
        # if the stacks are not equal after removing elements 1 by 1 then
return this message

def main():
    stack_input = user_input() # store the user input to the variable
stack
    equal_height(stack_input[0],stack_input[1],stack_input[2]) # call the
equal_height function and pass the 3 stacks as arguments
main() # call the main function
while 1: # loop until the user choose not to continue
    choice = input("\nContinue? Y or N?: ").upper() # ask the user if
he/she wants to continue
    if choice == "Y": # if the user choose to continue then ask for the
user input again
```

```
        os.system('clear') # clear the screen
        main() # call the main function again
    elif choice == "N":
        print(Fore.YELLOW+"\nThank you for using the
program."+Style.RESET_ALL)# if the user choose not to continue then exit
the program
        exit()
    else: # if the user input is not Y or N then ask again
        print(Fore.RED+"\nInvalid input"+Style.RESET_ALL)
```

TUTORIAL VIDEO

YouTube Link: https://youtu.be/T0qR6nsYs4k
My Github repository source code:
https://github.com/s0y4hh/DataStructureAndAlgorithm/blob/master/Stack_learningtask.py

TAKEAWAYS

After finishing the learning task, I got more and more learning on how to solve problems and about the stack. It's easy to implement using arrays but you need to keep in mind the rules on how the stack works and the problem which is to find the equal height of the stack. At first I couldn't instantly approach the problem because I needed to plan how to solve it and how to write a clean code. that's what I also remembered, don't approach or write a code first, plan and think how to write a code that will work as expected.